CLAIMS

What is claimed is:

1. A computer system, comprising:

a pipelined, simultaneous and redundantly threaded ("SRT") processor;

an I/O controller coupled to said processor, which in turn is coupled to at least one I/O device;

a main system memory coupled to said processor; and

a cycle counter configured to count clock cycles and advances once for each cycle of the processor clock;

wherein said SRT processor processes a set of instructions in a leading thread and also in a redundant trailing thread to detect transient faults in the computer system; and

wherein when a read cycle count command appears in the leading thread, the processor loads the current value of the cycle counter and replicates the value for the corresponding read cycle count command in the trailing thread.

2. The computer system of claim 1 further comprising a cycle count queue;

wherein when the processor loads the current value of the cycle counter, the processor stores the same value in a cycle count queue.

3. The computer system of claim 2 wherein the processor accesses the cycle count queue and not the cycle counter to load cycle count values in response to read cycle count instructions in the trailing thread.

4. The computer system of claim 2 wherein the read cycle queue is a FIFO buffer.

5. The computer system of claim 2 wherein the cycle count entries in the cycle count queue comprise a program count identifier and the cycle count value that was retrieved by the processor in response to the corresponding read cycle count command in the leading thread.

6. The computer system of claim 4 wherein all read cycle count commands in the leading and trailing threads are executed by the processor in their original, program order.

7. The computer system of claim 6 wherein if the cycle count queue becomes full, execution of instructions in the leading thread is temporary halted to prevent more cycle count values from entering the cycle count queue; and

wherein if the cycle count queue becomes empty, execution of instructions in the second thread is temporary halted to allow more cycle count values to enter the cycle count queue.

8. A pipelined, simultaneous and redundantly threaded ("SRT") processor, comprising:

a program counter configured to assign program count identifiers to instructions in each thread that are fetched by the processor;

a register update unit configured to store a queue of instructions prior to execution by the processor;

floating point execution units configured to execute floating point instructions;

integer execution units configured to execute integer-based instructions;

load/store units configured to perform load and store operations to or from data locations such as a data cache and data registers; and

a cycle counter configured to keep a running count of processor clock cycles;

wherein said processor is configured to detect transient faults during program execution by executing instructions in at least two redundant copies of a program thread and wherein false errors caused by incorrectly replicating cycle count values in the redundant program threads are avoided by using the actual values from cycle count reads in a first program thread for a second program thread.

9. The SRT processor of claim 8 wherein the processor further comprises:

a cycle count queue for storing the actual values fetched by read cycle count instructions in the first program thread;

wherein the load/store units place a duplicate copy of the cycle count value in the cycle count queue after fetching the cycle count value from the cycle counter.

10. The SRT processor of claim 9 wherein the load/store units access the cycle count queue and not the cycle counter to fetch cycle count values in response to read cycle count instructions in the second program thread.

11. The SRT processor of claim 10 wherein the SRT processor is an out-of-order processor capable of executing instructions in the most efficient order, but wherein read cycle count instructions are executed in the same order in both the first and second program threads.

12. The SRT processor of claim 11 wherein the cycle count queue is a FIFO buffer and data is transmitted to and from the buffer using an error correction technique.

13. The SRT processor of claim 12 wherein the individual cycle count values stored in the cycle count queue comprise:

a cycle count value that was returned by the corresponding read cycle count instruction in the leading thread.

14. The SRT processor of claim 12 wherein if the cycle count queue becomes full, the first thread is stalled to prevent more cycle count values from entering the cycle count queue; and

wherein if the cycle count queue becomes empty, the second thread is stalled to allow cycle count values to enter the cycle count queue.

15. The SRT processor of claim 11 wherein the register update unit is capable of managing program order for the read cycle count instructions by establishing a dependence with instructions before and after the read cycle count instructions.

16. A method of replicating cycle counter values in an SRT processor which can fetch and execute a set of instructions in two separate threads so that each thread includes substantially the same instructions as the other thread, one of said threads being a leading thread and the other of said threads being a trailing thread, the method comprising:

probing the cycle counter to fetch the current value of the cycle counter when the leading thread requests the cycle count;

storing the current value in a cycle counter queue;

probing the cycle counter queue for the cycle count value for corresponding cycle count requests in the trailing thread.

17. The method of claim 16 further comprising:

executing the cycle count requests in the leading and trailing threads in program order.

18. The method of claim 17 wherein the entries in the cycle count queue comprise a program count identifier and the cycle count value.

19. The method of claim 18 further comprising implementing a FIFO buffer as the cycle count queue.

20. The method of claim 19 wherein:

if the buffer becomes full, the leading thread is stalled to prevent more cycle counts from entering the buffer; and

wherein if the buffer becomes empty, the trailing thread is stalled to allow more cycle counts to enter the buffer.

21. The method of claim 18 further comprising:

transmitting data to and from the cycle count queue using an error correction technique.

22. A method of replicating cycle counter values in an SRT processor which can fetch and execute a set of instructions in two separate threads so that each thread includes substantially the same instructions as the other thread, one of said threads being a leading thread and the other of said threads being a trailing thread, the method comprising:

stalling execution of the leading thread when a read cycle count ("RCC") command is encountered in the leading thread;

executing instructions in the trailing thread until the corresponding RCC command is encountered in the leading thread; and

fetching a single copy of the cycle count value from the cycle counter and distributing said value to both threads.

23. The method of claim 22 further comprising:

maintaining a predetermined slack between execution of the leading and trailing threads during normal operation;

temporarily permitting the reduction of the predetermined slack to allow synchronization of the threads; and

resuming the predetermined slack after the RCC command is executed.